Growing Support Vector Classifiers via Architecture Boosting

Parrado-Hernández, E., Mora-Jiménez, I., and Navia-Vázquez, A.
Department of Communication Technologies, Universidad Carlos III de Madrid
Avda de la Universidad 30, 28911 Leganés-Madrid, Spain

Abstract—

It is widely recognized that one of the main advantages of Support Vector Machines over other training techniques, apart from their good generalization capabilities, is their ability to automatically obtain the machine architecture given a training data set. Unfortunately, this is achieved after solving a large optimization problem and, in many cases, the resulting machines can be exceedingly large (a lot of Support Vectors are obtained). In this paper we propose a method for incrementally growing Support Vector Classifiers (SVCs), the Growing SVC (GSVC) algorithm, such that, after starting with very simple machines, the classifier is gradually improved by concentrating on critical regions (hard to learn) of the input space, thereby asymptotically approaching the SVC solution. Every time the architecture is increased, only parameter update is needed, and not retraining.

The performance of the proposed training scheme will be illustrated using well known practical problems. Finally, some future lines of research are briefly mentioned.

I. Introduction

Support Vector Classifiers (SVCs) are able to find maximal margin boundaries between classes [24], and yield minimal structural risk solutions (good generalization capabilities). The training process is computationally costly (Quadratic Programming (QP) optimization), even when solved using highly efficient procedures [10][18]). Furthermore, SVCs are claimed to automatically obtain the machine architecture given a training data set, such that the classifier is expressed in terms of some of the input patterns (the so-called Support Vectors (SVs)), every one of them becoming the prototype of a kernel function. When the problem to solve involves many patterns with classes highly overlapped (situation very common in real world problems), the number of SVs found by this procedure is often exceedingly large (sometimes 1/4 or even 1/3 of the total amount of data), leading to very large machines (requiring hundreds or thousands of kernel computations to process every new pattern). Several procedures exist, though, to reduce the final complexity of the machine ([4][14][23]), but they are also complex, and they need the SVC solution as a starting point. In a previous work we have shown that this approach does not lead to good solutions [13]. We will adopt here a different strategy, instead of solving the whole problem and reducing the machine afterwards, we will grow the architecture until performance is no longer improved. We expect large savings in computational cost and machine complexity. Before presenting the growing scheme for SVCs, let us briefly review the main concepts of SVC learning.

Basically, nonlinear SVCs project training patterns $((\mathbf{x}_i, y_i), i = 1, ..., N)$ onto an infinite dimensional space F, $(\mathbf{x}_i \rightarrow \boldsymbol{\phi}(\mathbf{x}_i))$, where the maximal margin hyperplane is found $f(\mathbf{x}) = \mathbf{w}^T\boldsymbol{\phi}(\mathbf{x})$, yielding a decision machine $y = sign(f(\mathbf{x}))$. Unfortunately, in many interesting cases $\boldsymbol{\phi}()$ is unknown or the dimension of F is infinite, such that $\mathbf{w}$ can not be directly obtained. The usual approach is to solve the problem using its dual formulation in terms of inner products or kernels $K(\mathbf{x}_i, \mathbf{x}_j) = \boldsymbol{\phi}^T(\mathbf{x}_i)\boldsymbol{\phi}(\mathbf{x}_j)$, which leads to the complex QP minimization mentioned above. Nevertheless, there is an approximate (but effective) way for obtaining $\mathbf{w}$ using the primal formulation. Taking into account that N vectors always span a subspace of dimension (at most) N, we can express $\mathbf{w}$ as a linear combination of the projected data, the weights being[1] $\alpha_i y_i$ $(\mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \boldsymbol{\phi}(\mathbf{x}_i))$. Collecting all nonzero terms, we obtain the familiar expression for the classifier in terms of SVs $f(\mathbf{x}_j) = \mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_j) = \sum_{i=1}^{N_s} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}_j)$. Furthermore, these N vectors usually lie in a (lower) R-dimensional space[2], and we can finally adopt an even more simplified expression using only R weights to specify the hyperplane:

$$f(\mathbf{x}) \simeq \mathbf{w}_R^T(\boldsymbol{\Phi}_R^T\boldsymbol{\phi}(\mathbf{x})) = \sum_{i=1}^{R} w_{R,i} K(\mathbf{c}_i, \mathbf{x}) \qquad (1)$$

where $\boldsymbol{\Phi}_R = [\boldsymbol{\phi}(\mathbf{c}_1), \boldsymbol{\phi}(\mathbf{c}_2), ..., \boldsymbol{\phi}(\mathbf{c}_R)]$ is the corresponding base, every representational axis $\boldsymbol{\phi}(\mathbf{c}_i)$ can be associated to patterns $\mathbf{c}_i$ in input space (centroids in the Radial Basis Functions (RBF) case). In [13] we proposed several methods (such as nonlinear Principal Component Analysis (PCA), or clustering techniques) to select these $\mathbf{c}_i$ values, and we have shown that efficient Weighted Least Squares (WLS) methods can be applied to solve the SVC problem under this formulation, by iteratively computing weighting values a_i and solving WLS problems[3] involving matrices of size $R \times R$. Alternative (non-parametric) implementations of this iterated WLS procedure can be found in [15][16]. This WLS-SVC algorithm has proved to be computationally very efficient (1-2 orders of magnitude speed-up with

[1] Many of these values are zero (non Support values), which reduces the size of the machine.

[2] The approximate dimension R of projected data can be obtained using nonlinear PCA as in [22], but we show in [13] that results are not very sensitive to this number, i.e., slightly larger values can be used without appreciable degradation.

[3] For further details, please, refer to [13]

respect to highly optimized techniques such as chunk-SVC [10] or Sequential Minimal Optimization (SMO) [18]), and leads to lower complexity machines (also 1-2 orders of magnitude), for a similar performance in terms of Classification Error (CE) [13]. As in QP-SVC, we only need to handle kernel computations during training. We will denote this training process as $(\mathbf{w}, \mathbf{a}) = WLSSVC(\mathbf{K}_R, y_i)$, where $\mathbf{K}_R$ ($\{\mathbf{K}_R\}_{j,k} = K(\mathbf{x}_j, \mathbf{c}_k)$) is the kernel matrix for the entire dataset. Applying this WLS-SVC algorithm we obtain the value for $\mathbf{w}$ as well as weighting values a_i, which vanish for non-SV patterns. In what follows, we propose to build an asymptotical approximation to the SVC solution starting with very simple machines and selecting new elements $\boldsymbol{\phi}(\mathbf{c}_i)$ in conflictive regions of the space, i.e., where data is not being correctly classified and it is being poorly represented with the current representational base in F, as explained in the next Section.

II. Growing Support Vector Classifiers

Using the formulation in the previous section, it is straightforward to set up a growing procedure for SVCs. We start with a very reduced machine, which will provide us with the basic guidelines to continue expanding the architecture. As also shown in [13], sensitivity with respect to representational axis selection (centroids in input space) is very low, and we propose to use the least effort in this task: choose $\mathbf{c}_i$ by picking up at random $M/2$ patterns of every class. We can compute the initial kernel matrix $\mathbf{K}_0$ and obtain the initial hyperplane $\mathbf{w}(0)$ and weighting values $\mathbf{a}(0)$ via WLS-SVC. With so few axes in F, the representation of the dataset and the performance of the classifier are very poor. Nevertheless, this preliminary information can be used to select new axes to increase the representational capabilites of the architecture, to join them to the existing ones and update the SVC solution. The architecture growing is continued until a convergence criterion is reached. A schematic representation of the growing architecture is provided in Figure 1 below, and the procedure can be summarized in the steps displayed in Table II.

Note that, every time the classifier is expanded with new units, the GSVC algorithm only updates the parameters for step "n" using those at step "n-1", such that does not need re-training from scratch. This is an important characteristic of the algorithm, which is feasible due to the iterative nature of WLS-SVC training, and that would we impossible to implement under a QP formulation. Furthermore, all the kernels computed at step "n" do not need to be recomputed at following steps. Finally, and most important, although we are using a (boosting-like) suboptimal procedure for growing the architecture, weights are globally optimized, in the sense that structural risk minimization is preserved, gradually approaching the QP-SVC solution. The proposed criterion for selecting new centroids will be discussed in the following section.

III. Representational axis selection criteria

It has been shown that good procedures exist to find before training a near-optimal subset of representational axis

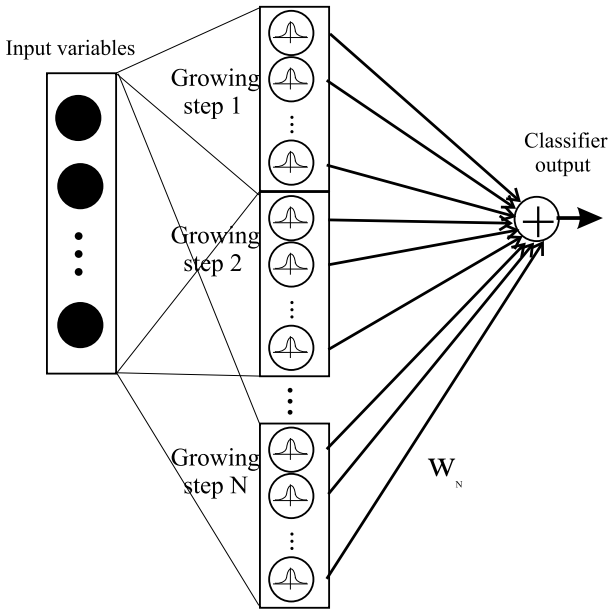

Fig. 1. Architecture growing via boosting.

$\boldsymbol{\phi}_i$ in F for a given dataset, by means of nonlinear PCA [23], such that they define an orthonormal basis. Unfortunately, proceeding this way is difficult. Firstly, we need to find the Singular Value Decomposition (SVD) of a matrix of size $N \times N$, to obtain the new axes $\boldsymbol{\phi}_i$, which is a very costly process even for moderate values of N. Secondly, we need to find elements $\mathbf{c}_i$ in the input space such that $\boldsymbol{\phi}(\mathbf{c}_i) \simeq \boldsymbol{\phi}_i$ to obtain a final machine of the form (1). Although this procedure has already been formulated as an Expectation Maximization (EM) algorithm [23], its correct application is not a trivial task. When this problem is stated as a clustering in input space ([23], [13]), it requires to set R 'a priori', and solve a large clustering problem, and eventually many of the centroids might be spent in modeling regions with little interest.

We propose to find a good set of representational axes with a procedure with the same principles of Boosting [7]. For a given architecture (step "n"), we force the training procedure to concentrate on conflictive patterns, thereby improving the representation of this region of the input space. By conflictive we mean missclassified or "close to the boundary", and SVC provides us with an automatic criterion to identify those patterns: the margin and support vector concepts. SVC training (even in its WLS-SVC implementation) provides a list of those patterns critical for training (SVs) (patterns with $a_i > 0$, under WLS-SVC scheme). Hopefully, by concentrating on these hard-to-learn patterns, the overall performance of the machine can be improved, since they represent a sufficient set of data for solving the problem at hand. Nevertheless, we know that, in problems with highly overlapped classes, a large number of SVs can be found in the overlapping areas. Therefore, an additional criterion has to be used to avoid selecting a lot of patterns in the same region. We propose the following combined criterion: among those patterns with $a_i > 0$,

TABLE I

SUMMARIZATION OF THE GSVC ALGORITHM.

0. Initialization
- Select M patterns at random ($M/2$ in every class),
- build initial kernel matrix $\mathbf{K}_0$ and
- obtain the initial hyperplane $(\mathbf{w}(0), \mathbf{a}(0)) = WLSSVC(\mathbf{K}_0, y_i)$
Loop n=1,2,...
1. Find a new set of centroids to expand the architecture, compute $\mathbf{K}_n$ and build $\mathbf{K}(n) = [\mathbf{K}_1, ..., \mathbf{K}_n]$
2. Train the new architecture, by updating weights using WLS-SVC
$(\mathbf{w}(n), \mathbf{a}(n)) = WLSSVC(\mathbf{K}(n), y_i)$
3. Evaluate the train and validation sets, and decide to continue growing or not.

choose randomly between the ones with smaller maximal projection onto the actual base. This quantity can be easily estimated using the kernel matrix, by simply finding the maximum of the kernel values associated to every pattern, since every kernel itself represents the projection onto one axis.

Every time the architecture is incremented, weight parameter w_N is updated using the global SVC criterion by means of WLS-SVC, such that, although a suboptimally growing procedure is proposed for the architecture, the global performance of the classifier is controlled by a well theoretically founded criterion for structural risk minimization, that provided by SVM theory.

IV. EXPERIMENTS

We will present in this Section several experiments to illustrate the capabilities of the proposed GSVC algorithm. We will show firstly its convergence behaviour, by depicting the decision boundaries found at several stages of training. Then we compare the performance of GSVC using several well known problems with those obtained with QP-SVC, Gaussian Art Map (GAM) [25], and C4.5 [20] algorithms. For those separable problems ($CE = 0$ on the test set), we depict the decission boundaries found by every method.

A. Convergence behaviour and complexity reduction

We solve here a simple 2D classification problem with synthetic data generated with a mixture of Gaussian and uniform distributions placed at random. The objective is not to compare performance, but to illustrate the operation of the GSVC algorithm. In Figure 2 we have represented the incremental approach to the SVC solution provided by the GSVC algorithm (boundaries found at steps 1, 2 and 10 in (a), (b) and (c), respectively). Training patterns have been represented using "+" and "o" for every class, and the decision boundary (continuous line) and margins (dashed line) are shown.

Classification error (in percentage) in train, validation and test sets have been represented in Figure 2(d), it can be observed how, in spite of the continuous growth in complexity of the classifier (4 new units are added in every step), the underlying structural risk minimization principle guarantees that overfitting does not occur. In this sense, any stopping criteria that detects the paralization in convergence observed after step 8 is valid to stop the training process.

B. Comparison with other classifiers

We will compare in this Section the performance, training cost and final machine size of GSVC algorithm with those of QP-SVC[4], GAM and C4.5. We have selected several well known binary classification problems, and simulation results have been collected in Table II. Parameters have been selected to give the best results on the validation set, and are also detailed in the same table. The problem set includes the following tasks:

- Classify the patterns lying inside a circle centered on a unit square (*Circle* in Table II).
- Classify the patterns belonging to two different nested spirals (*Spiral* in Table II).
- Classify the patterns belonging to two different overlapped Gaussian distributions (*Two Gauss* in Table II).
- Classify the Adult dataset of UCI repository [2] (*Adult* in Table II).
- Classify the odd and even numbers in the Optical Recognition of Handwritten Digits dataset of UCI repository (*Hand-dig* in Table II).

The criteria used in the comparison are the Clasification Error (*Test CE* in Table II), the Machine Complexity (*Mach. size*) measured as the number of RBFs in GSVC, SVMlight and GAM and the number of nodes of the decision tree generated by C4.5; and the Cost of the Training Algorithm (*Train cost*), measured in CPU seconds.

In order to qualitatively evaluate the boundary found in those 2D cases with very similar CE (zero in some cases), we have depicted in Figure 3 the boundaries found with GSVC, QP-SVC, GAM and C4.5 for the circle, spiral and Gaussian cases.

The results of the comparison between GSVC and SVMlight show a spectacular reduction in the complexity of the machine achieved by GSVC. The reduction rate reaches

[4]For the software implementation of the original SVC, SVMlight algorithm has been used

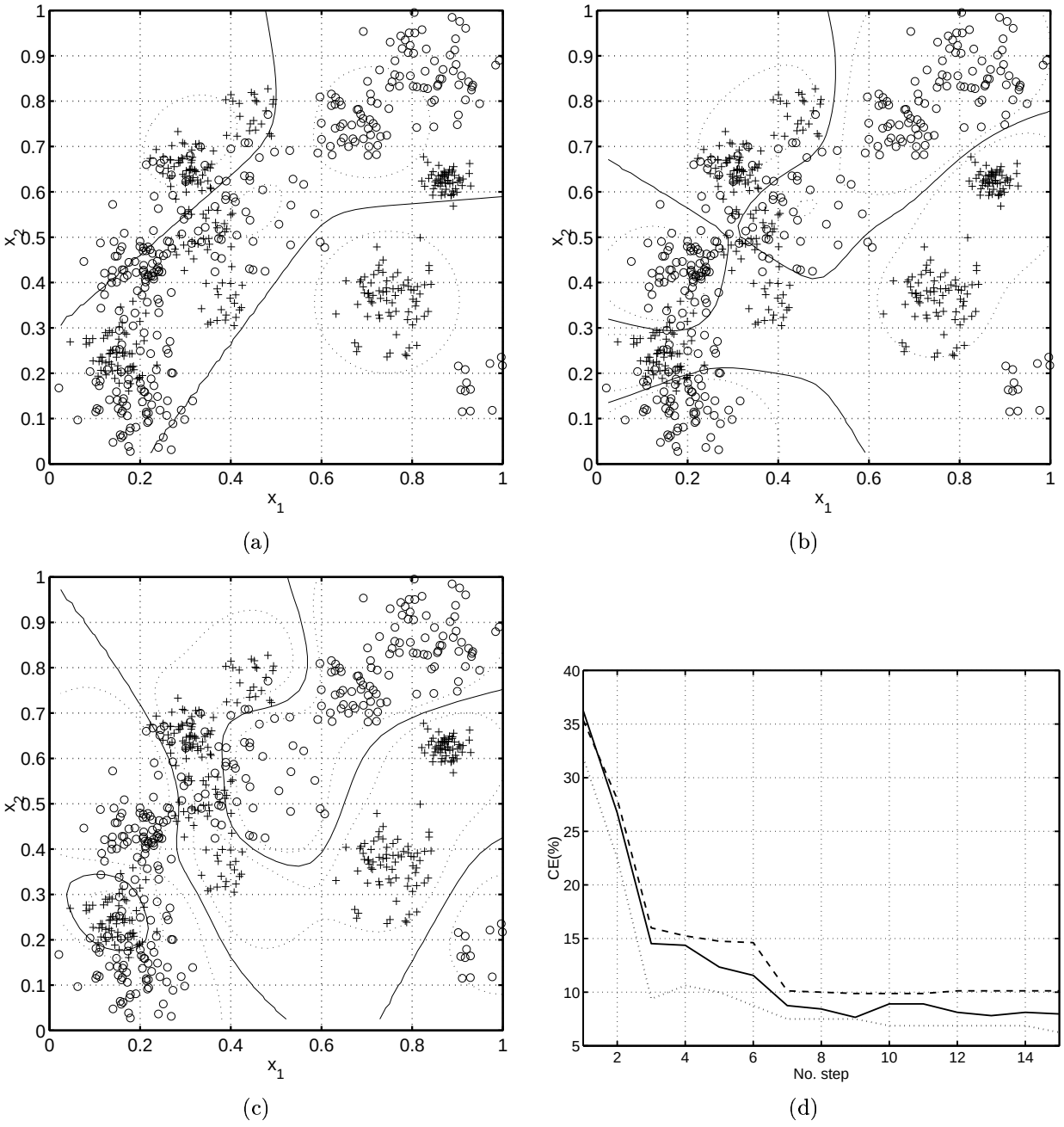

Fig. 2. Decision boundaries (continuous line) and margin (dashed line) at steps 1,3 and 10 of the GSVC algorithm, in (a), (b) and (c), respectively. Convergence error for train (continuous line), validation (dotted) and test (dashed) is shown in (d).

even 98% of the SVMlight final size. Moreover, classification error is not affected by this reduction in the size of the machine, since both algorithms present similar accuracies in 4 of the 5 tasks.

With respect to GAM, GSVC achieves better classification accuracy with machines of the same number of nodes. Although for the simpler tasks (circle and spiral) GAM training cost measured in CPU seconds is smaller than GSVC one, for the more difficult problems either both costs are similar (adult) or GSVC training is faster (Gauss and hand-dig).

Finally, GSVC has been found to reduce the complexity of the machines with relation to those obtained by C4.5 in the most difficult tasks, while its classification error is smaller in 4 of the 5 tasks.

V. Conclusions and future work

We have presented an incremental procedure for SVC training relying on Iterated Weighted Least Squares minimizations, the proposed GSVC algorithm presenting several main benefits. Firstly, the good generalization capabilities of SVC theory are preserved, since we are also carrying out an structural risk minimization in every step. Secondly,

TABLE II

RESULTS OF THE COMPARISON OF THE FOUR CLASSIFIERS. FOR EACH ALGORITHM, CLASSIFICATION ERROR (TEST CE), MACHINE COMPLEXITY (MACH. SIZE) AND TRAINING COST (TRAIN COST) IN CPU SECONDS ARE DISPLAYED. DESIGN PARAMETERS FOR GSVC ARE σ, SIZE OF THE RBF, AND M, NUMBER OF NEW CENTROIDS ADDED TO THE ARCHITECTURE IN EACH ITERATION; FOR SVMLIGHT σ IS ALSO THE SIZE OF THE RBF; FOR GAM, γ IS THE INITIAL SIZE OF EACH RBF AND ρ IS THE BASELINE VIGILANCE PARAMETER; FOR C4.5, I_{conf} IS THE INTERVAL OF CONFIDENCE USED FOR PRUNING THE TREE.

		Circle	Spiral	Two Gauss	Adult	Hand-dig
GSVC	Test CE(%)	0	0	17.9	15.3	4.06
	Mach. size	12	60	20	120	90
	Train cost	9.24	15.30	6.93	2892	590.38
	Parameters	$\sigma = 0.5$, $M = 2$	$\sigma = 0.25$, $M = 4$	$\sigma = 2$, $M = 2$	$\sigma = 5$, $M = 6$	$\sigma = 20$, $M = 8$
SVM light	Test CE(%)	0	0	17.3	15.35	2.00
	Mach. size	17	106	403	10358	273
	Train cost	1.19	133	1426	2633	68.59
	Parameters	$\sigma = 0.5$	$\sigma = 0.25$	$\sigma = 2$	$\sigma = 5$	$\sigma = 20$
GAM	Test CE(%)	0	2.20	16.90	20.82	7.95
	Mach. size	6	20	35	131	46
	Train cost	0.06	2.55	11.59	2582	1364
	Parameters	$\gamma = 0.5$, $\rho = 10^{-14}$	$\gamma = 1$, $\rho = 10^{-14}$	$\gamma = 2$, $\rho = 10^{-14}$	$\gamma = 2$, $\rho = 10^{-231}$	$\gamma = 1$, $\rho = 10^{-448}$
C4.5	Test CE(%)	0	43.3	18.4	14.5	7.06
	Mach. size	9	3	11	469	157
	Train cost	0.06	0.02	0.07	25.83	4.94
	Parameters	$I_{conf} = 5\%$	$I_{conf} = 5\%$	$I_{conf} = 5\%$	$I_{conf} = 15\%$	$I_{conf} = 25\%$

the resulting machine is usually much smaller than those found by QP-SVC, dramatically simplifying the operation of the classifier.

In the near future, an optimal pruning stage will be introduced in the GSVC algorithm in order to simplify the architecture of the machine even more. Since we have proposed a growing procedure for growing SVCs based on some very computationally simple criteria (random selection of new centroids in a critical set), the structure obtained after several growing steps may be redundant to some extent. It is possible to carry out a nonlinear PCA analysis to prune those superfluous nodes, analogously as proposed in [23].

Furthermore, we propose to combine the present growing scheme with the adaptive implementation of SVCs introduced in a previous work, and to evaluate the possibility of combining kernels with different parameters in the same classifier, controlling during the training process the selection of these parameters by cross-validation.

REFERENCES

[1] S. Asogawa, and N. Akamatsu, "An efficient algorithm for clustering data and best model selection using AIC," In *Proc. IEEE Int. Conf. on Neural Networks*, vol 1, pp. 540-545, 1994.

[2] C.L.Blake, and C.J. Merz, UCI Repository of machine learning databases Irvine, CA: University of California, Department of Information and Computer Science, 1998. [http://www.ics.uci.edu/~mlearn/MLRepository.html].

[3] M. Bonneville, J. Meunier, Y. Bengio, and J.P. Soucy, "Support vector machines for improving the classification of brain PET images," In *Proc. SPIE, The International Society for Optical Engineering*, vol. 3338, pp. 264-273, 1998.

[4] C.J.C. Burges and B. Scholkopf, "Improving the accuracy and speed of support vector machines," In M. Mozer, M. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems 9*, pp. 375–381. MIT Press, Cambridge, CA, 1997.

[5] C.J.C. Burges, "A tutorial on Support vector machines for pattern recognition," *Data Mining and knowledge discovery*, vol. 2, pp. 121-167, 1998.

[6] S. Chen, S. Gibson, G.J. Cowan, and P.M. Grant, "Reconstruction of binary signals using an adaptive radial-basis-function equalizer," *Signal Processing*, vol. 22, pp. 77–93, 1991.

[7] Y. Freund, and R.E. Schapire, "A decisison-theoretic generalization of on-line learning and an application to boosting," In *Proc. of the 2nd Annual European Conf. on COmputational Learning Theory*, 1995.

[8] J. Huang, X. Shao, and H. Wechsler, "Face pose discrimination using support vector machines (SVM)," In *Proc. 14th Int. Conf. on Pattern Recognition*, vol. 1, pp. 154-156, 1998.

[9] A.K. Jain and R.C. Rubes, *Algorithms for Clustering Data*. New York: Prentice-Hall Advanced Series, 1980.

[10] T. Joachims, "Text categorization with support vector machines: learning with many relevant features," In *Proc. ECML European Conference on Machine Learning*, vol. 1, pp. 137-142, 1998.

[11] T. Joachims, "Making large-Scale SVM Learning Practical," In B. Schlkopf, C. Burges and A. Smola, *Advances in Kernel Methods - Support Vector Learning*, pp. 169–184. MIT Press, Cambridge, CA, 1999.

[12] Y. Linde, A. Buzo, and R.M. Gray, "An algorithm for vector quantizer design," *IEEE Trans. Commun.*, vol. 28, pp. 84-95, 1980.

[13] A. Navia-Vázquez, F. Pérez-Cruz, A. Artés-Rodríguez, and A.R. Figueiras-Vidal. "Weighted Least Squares Training of Support Vector Classifiers leading to Compact and Adaptive Schemes," *Submitted to IEEE Trans. Neural Networks.*, 1999.

[14] E. Osuna, and F. Girosi, "Reducing the run-time complexity of Support Vector Machines," In B. Scholkopf, C.J.C. Burges, and A.J. Smola, eds. *Advances in Kernel Methods: Support Vector Learning*, pp. 271-284, MIT Press: Cambridge, MA, 1999.

[15] F. Pérez-Cruz, A. Navia-Vázquez, J.L. Rojo-Alvarez, A. Artés-Rodríguez, "A New Training Algorithm for Support Vector Machines," In Proceedings of the Fifth Bayona Workshop on Emerging Technologies in Telecommunications, Bayona, Spain, 1999.

[16] F. Pérez-Cruz, A. Navia-Vázquez, P. Alarcón-Diana, A. Artés-Rodríguez, "Support Vector Classifier With Hyperbolic Tangent Penalty Function," In Proceedings of ICASSP 2000, 2000.

[17] F. Pérez-Cruz, A. Navia-Vázquez, P. Alarcón-Diana, A. Artés-Rodríguez, "An IRLS Procedure for SVR," Accepted for publication at EUSIPCO 2000.

[18] J. Platt, "Fast training of support vector machines using sequential minimal optimization," In B. Scholkopf, C.J.C. Burges, and A.J. Smola, eds. *Advances in Kernel Methods: Support Vector Learning*, pp. 185-208, MIT Press: Cambridge, MA, 1999.

[19] M. Prakash, and M.N. Murty, "A genetic approach for selection

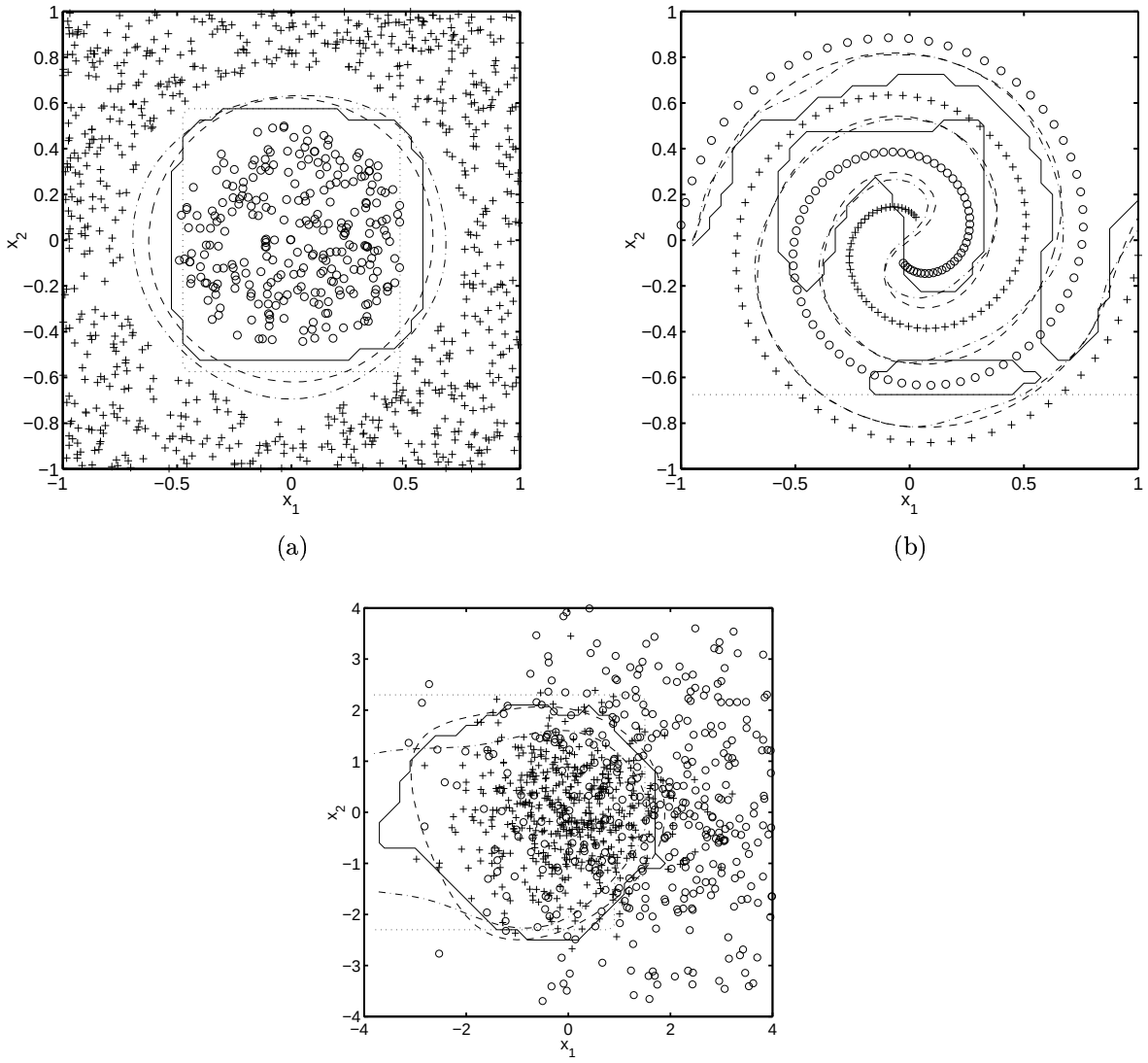

Fig. 3. Decision boundaries obtained with the four classifier algorithms. GSVC boundary corresponds to the dash-dotted line; SVMlight to the dashed line; GAM to the solid line and C4.5 to the dotted line.

of (near-)optimal subsets of principal components for discrimination," *Pattern Recognition Letters*, vol. 16, pp. 781–787, 1995.

[20] J.R. Quinlan, "C4.5: Programs for Machine Learning", Morgan Kaufmann, San Mateo, 1993.

[21] B.D. Ripley, "Neural networks and related methods for classification (with discussion)," *J. Royal Statistical Soc. Series B*, vol. 56, pp. 409–456, 1994.

[22] B. Scholkopf, "Nonlinear component analysis in a kernel eigenvalue problem," *Neural Computation*, vol. 10, pp. 1299–1319, 1998.

[23] B. Scholkopf, P. Knirsch, A. Smola, and C. Burges, "Fast approximation of support vector kernel expansions, and an interpretation of clustering as approximation in feature spaces," In *Proc. 20. DAGM Symposium Mustererkennung, Springer Lecture Notes in Computer Science*, vol. 1, 1998.

[24] V. Vapnik, *The Nature of Statistical Learning Theory*. Springer-Verlag, New York, 1995.

[25] J.R. Williamson, "A Constructive, Incremental-Learning Network for Mixture Modeling and Classification," *Neural Computation*, vol. 9, pp. 1517–1543, 1997.